MythX

REPORT 5F6ACEA85B4F3D00182AFCB3

Created	Wed Sep 23 2020 04:27:20 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	kohshi.shiba@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
8fbedfde-bbe9-4a16-ae09-90bc1bb7c1f9	/contracts/connector.sol	7

Analysis 8fbedfde-bbe9-4a16-ae09-90bc1bb7c1f9 MythX

Started	Wed Sep 23 2020 04:27:25 GMT+0000 (Coordinated Universal Time)
Finished	Wed Sep 23 2020 05:12:36 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Connector.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	2	5

ISSUES

MEDIUM Loop over unbounded data structure.

SWC-128

Gas consumption in function "settleMarket" in contract "Connector" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Source file

/contracts/connector.sol

Locations

```
87 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
88 ) {
89 for (uint256 i = 0; i < length - 1; i++) {
90 payout[i] = 0;
91 }
```

MEDIUM Loop over unbounded data structure.

SWC-128

Gas consumption in function "settleMarket" in contract "Connector" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Source file

/contracts/connector.sol

Locations

```
93 } else {
94 uint256 result = uint256(response);
95 for (uint256 i = 0; i < length; i++) {
96 if (i != result) {
97 payout[i] = 0;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/contracts/connector.sol

Locations

```
5 */
6
7 pragma solidity ^0.6.0;
8 pragma experimental ABIEncoderV2;
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "factory" is internal. Other possible visibility settings are public and private.

Source file

/contracts/connector.sol

Locations

```
17 using Address for address;
18
19 address factory;
20 address reality;
21 address arbitrator;
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "reality" is internal. Other possible visibility settings are public and private.

Source file

/contracts/connector.sol

Locations

```
18
19 address factory;
20 address reality;
21 address arbitrator;
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "arbitrator" is internal. Other possible visibility settings are public and private.

Source file

/contracts/connector.sol

Locations

```
19 address factory;
20 address reality;
21 address arbitrator;
22
23 mapping(address => bytes32) questionId;
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "questionId" is internal. Other possible visibility settings are public and private.

Source file

/contracts/connector.sol

Locations

```
21 address arbitrator;
22
23 mapping(address => bytes32) questionId;
24
25 constructor(
```